1 Introduction

1.1 What is a system?

Electrical and computer engineers love to think in terms of signals and systems. Loosely speaking, a signal is something that carries information, and a system is something that transforms one type of signal into another. Signals are described by functions (usually of time and/or space, but not always), and systems are described by more sophisticated functions that act on functions to produce other functions. This viewpoint is broader and more abstract than what you may have seen in courses like ECE 210 (Analog Signal Processing) or ECE 310 (Digital Signal Processing), but it is actually very natural, and one of the purposes of this course is to get you to think about signals and systems in this manner. Let us see some examples:

- A digital camera is a system that relies on a combination of optical and electronic components to transform light that hits the lense during the exposure period into a digital image.
- A computer network is a system that relies on a combination of routers, switches, and communication channels to deliver content (text, images, audio, video, etc.) from a set of sources to a set of destinations. The content traverses the network in the form of packet streams.
- A piece of software is a system that transforms user's input into a file that can be stored on the computer or into a data stream that can be transmitted to some output device. For example, the mp3 decoder takes an `.mp3` file and generates an audio signal.
- The stock market is a system that transforms stock prices at the beginning of a given period (say, a day) to stock prices at the end of the period.

In this way, you can probably come up with lots of other interesting examples.

1.2 OK, but what is a *stochastic* system?

So, at this point, a system is some function that operates on signals to produce other signals. In order to speak about stochastic systems, we must define what *stochastic* means. If we search for the definition of "stochastic" on Google, we get this:

> **stochastic** (adj.) — randomly determined; having a random probability distribution or pattern that may be analyzed statistically but may not be predicted precisely.

So, a stochastic signal is a signal that cannot be specified exactly because of some chance effects. For example, suppose we look at an electric circuit and monitor the voltage across a resistor over a period of time. Resistors tend to heat up, and heat is actually a manifestation of random thermal motion of molecules. Because of this thermal motion, there will be random fluctuations in the voltage across the resistor, so this voltage is a *stochastic* function of time. We call such functions *stochastic processes*, where the use of the word "process" is intended to evoke something that evolves (usually in time). We will develop formal machinery to describe and model stochastic processes, but for now we can just say that the stochastic aspect of the voltage waveform is that, even if we know the value of the voltage at some time t, we cannot be certain about its value at some later time $t' > t$. All we can do is assign a *probability distribution* to the possible values of the voltage at time t' given its value at time t. However, this is already something that we can work with, and in particular we should be able to get answers to questions like

- what is the *average* or (*mean* or *expected*) value of the voltage at time t', given what we know about the voltage at time t?
- what is the *variance* of the voltage at time t' — i.e., how much can we expect the actual voltage at time t' to fluctuate around this mean value?
- what is the probability that the voltage at time t' will rise above or fall below some threshold value?

We may also wonder about the *correlation* between the voltage at time t and the voltage at time t', there are many possibilities which we will explore in due time. The upshot is that, even though we may no longer have an exact description of the signal of interest, we may be able to analyze it *statistically* in terms of mean values, variances, correlation coefficients, etc.

Now we have at least an intuitive inkling of what constitutes a stochastic signal. What about a stochastic *system*? Recall that we have defined a system as a function that maps signals to signals. In the *deterministic* (i.e., nonstochastic) setting, if we know the operation of a system, we can give (at least in principle) an exact description of the output signal for a given input signal. By contrast, a *stochastic* system is any system, which maps a deterministic input signal into a stochastic output signal. Thus, for a stochastic system, even if we are given the exact description of the input signal, we can only make probabilistic statements about the output signal. For example, a simple model of a stochastic system is a noisy communication channel: the sender transmits a bit, which may get flipped en route to the receiver with some small probability p. Suppose we transmit 0. In the ideal (deterministic) case, we would know right away that the received bit would also be 0. But, in the presence of channel noise, this is no longer the case — the received bit is now a random quantity:

$$\textsc{SentBit} = 0 \qquad \longrightarrow \qquad \textsc{ReceivedBit} = \begin{cases} 0, & \text{with probability } 1-p \\ 1, & \text{with probability } p. \end{cases} \tag{1.1}$$

By the same token,

$$\textsc{SentBit} = 1 \qquad \longrightarrow \qquad \textsc{ReceivedBit} = \begin{cases} 1, & \text{with probability } 1-p \\ 0, & \text{with probability } p. \end{cases} \tag{1.2}$$

So, the noisy communication channel is a (very simple) stochastic system: it takes a deterministic input signal and generates a stochastic output signal.

So, our interest in this course lies with systems that take stochastic input signals and transform them into stochastic output signals. Courses like ECE 210/310 provide you with a toolbox for analyzing deterministic signals and systems, and you have gotten your first exposure to probability in ECE 313. In this course, we will see how the systems toolbox and the probabilistic toolbox can be seamlessly integrated to let us describe, analyze, and synthesize stochastic systems.

1.3 Noise, uncertainty, and randomness

The range of stochastic phenomena in science and engineering is truly vast, and we will be able to get only a glimpse of it. However, it is important to understand different sources and origins of stochastic behavior. For our purposes, it will be convenient to distinguish the following three sources:

Noise. When we speak of noise, we refer to various stochastic effects that are not under our control. The thermal noise that results in fluctuations of the voltage across a resistor is an example. Noisy transmission of a bit is another example. We will look at various models of noise in electronic devices.

Uncertainty. Uncertainty refers to stochastic effects that arise due to lack of information. There are many situations where one is dealing with a very complex system consisting of a large number of interacting subsystems, and, even though we may have a precise description of what every subsystem does in isolation, describing the overall system precisely may be too daunting of a task. In that case, we may settle for a probabilistic description instead. This viewpoint should sound familiar to anyone who has taken a course on statistical physics: Imagine we have N molecules of a gas in some finite volume. The motion of each molecule is subject to Newton's laws, so, in principle, we could write down a huge system of differential equations describing the behavior of the gas at any time t (e.g., the position and momentum of every molecule at every time). However, if the number of molecules N is very large (and it typically is!), writing down all these equations (let alone understanding what they say about the physics) is a hopeless affair. Instead, we settle for a probabilistic description and ask questions like "what is the probability distribution of the energies of the molecules?" This philosophy turns out to be very useful in engineering as well, where it can be applied to model complex communication networks, swarms of robots, nanomaterials, etc.

Randomness. In this course, we will use the term "randomness" to refer to any stochastic effects that are deliberately added to a system, using a random number generator or sometimes even a physical source of noise. For example, in machine learning applications we may need to access a massive array of data, but cannot afford to sweep through the entire arrray due to time or memory limitations. Instead, we may opt to access a small number of randomly chosen elements of the array. Another use of randomization is to prevent instabilities or undesirable events from happening. For example, it could happen that several packets arrive at a router simultaneously, causing a collision. In that case, the routing protocol may add small random delays to each packet's timestamp to resolve the collision.

Of course, the boundaries between these three sources of stochastic effects are rather fuzzy, and we may often spot all three sources in the context of the same system. Nevertheless, it is important to be aware of these distinctions as time goes on.

1.4 Descriptions of stochastic systems: declarative vs. imperative

As noted earlier, a stochastic system is any entity that transforms deterministic signals into stochastic signals. In order to analyze and design such systems, we need to have at our disposal a way of describing them. Just as with deterministic systems, there are two distinct ways of describing stochastic systems: *declarative* (what does the system do?) and *imperative* (how to implement such a system?).

To see the difference between these two types of description, let us recall the above example of a noisy communication channel — we transmit a single binary message (a bit), and the received bit is equal to the sent bit with probability $1-p$. In that case, Eqs. (1.1) and (1.2) constitute the declarative description. They tell us *what* will happen to the sent bit, but not *how* it will happen. The "how" is the province of imperative descriptions. For a given system, there can be multiple imperative descriptions that are all *functionally equivalent* — i.e., for our communication channel example, they should all model the behavior described by the declarative model in Eqs. (1.1) and (1.2). Let us see one such description. We will give it in the form of a *function* that accepts a binary input SentBit and returns a *stochastic* binary output ReceivedBit:

```
$x \leftarrow$ SentBit
$Z \leftarrow$ FlipCoin($p$)
ReceivedBit $\leftarrow x \oplus Z$
```

It is useful to think about such imperative descriptions in terms of computer code. According to the above description, our system takes the input SentBit, stores it in a binary variable x, then calls some function FlipCoin(p), stores the output of that function in a binary variable Z, and then returns ReceivedBit $= x \oplus Z$, the Boolean xor of x and Z.

Important! Note that we have used a lowercase letter x for the input and an uppercase letter Z for the coin flip outcome. This is because we view the input x as *deterministic*, whereas the internal variable Z is *stochastic*, and thus does not have a definite value. We will always adhere to this convention and use uppercase letters for stochastic quantities and lowercase letters for deterministic quantities.

We have not yet specified how the function FlipCoin works, but for now let us just take it for granted that it generates a random variable that takes value 1 with probability p and 0 with probability $1-p$. In other words, FlipCoin(p) generates a Bernoulli random variable with bias p. Let us prove that the above imperative description is functionally equivalent to the declarative description in (1.1) and (1.2). To that end, we need to compute the probability of error:

$$\begin{aligned}\mathbf{P}[\textsc{ReceivedBit} \neq \textsc{SentBit}] &= \mathbf{P}[x \oplus Z \neq x] \\ &= \mathbf{P}[x \oplus Z \neq x, Z = 0] + \mathbf{P}[x \oplus Z \neq x, Z = 1] \\ &= (1-p)\mathbf{P}[x \oplus 0 \neq x] + p\mathbf{P}[x \oplus 1 \neq x].\end{aligned}$$

Recall the function table for XOR: for any binary variable b, we have $b \oplus 0 = b$ and $b \oplus 1 = \bar{b}$, where the bar denotes Boolean NOT. Therefore, $\mathbf{P}[x \oplus 0 \neq x] = 0$ and $\mathbf{P}[x \oplus 1 \neq x] = 1$, and we conclude that, indeed,

$$\textsc{ReceivedBit} = \begin{cases} \textsc{SentBit}, & \text{with probability } 1-p \\ \textsc{SentBit} \oplus 1, & \text{with probability } p \end{cases}.$$

So, indeed, our imperative description is functionally equivalent to the declarative description. It remains to specify the function CoinFlip(p). To that end, we will assume that we have access to another function UNIFORM(a, b) that generates a random variable uniformly distributed on the unit interval $[a, b]$. For example, in MATLAB you would use the function `rand` to do this. So, here goes:

```
$U \leftarrow$ UNIFORM$(0,1)$
if $0 \leq U < p$
   COINFLIP$(p) \leftarrow 0$
else
   COINFLIP$(p) \leftarrow 1$
```

Exercise: Prove that this indeed implements COINFLIP(p).